Многонишкова библиотека в User и Kernel-Space режим

Христо Вълчанов

An User and Kernel Space Multi-threaded Library: *Developing of technology and the large range of possibilities offered by modern hardware allows the use of specialized high-performance approaches for the implementation of various software systems and algorithms. One of the most used and effective approach is the development of multithreaded software running in parallel on multiple processors. This paper presents the features of the implementation of multi-threaded library under Linux, which allows running both in user and kernel-space mode.*

Key words: *Multi-threaded library, Threads, User-space, Kernel-space.*

ВЪВЕДЕНИЕ

Развитието на технологиите и големия набор от възможности предлагани от съвременните хардуерни компоненти позволява използването на специализирани високопроизводителни подходи при реализацията на различните програмни системи и алгоритми. Един от най-използваните и ефективни такива подходи е създаването на многонишков софтуер работещ паралелно върху множество процесори. Актуалността на този тип задачи и нуждата от нови и съвременни подобрени разработки се аргументира все повече с появата на потребителски ориентирани процесори, съдържащи множество самостоятелни ядра (към момента 6-8), които могат да работят независимо едно от друго [3].

Актуална тенденция в момента е навлизането на многонишкови библиотеки като основна част от най-използваните и ефективни програмни езици като C++, Java и др. Множеството различни специфични архитектури прави тази задача трудна и дори на моменти невъзможна за ефективна реализация. Затова съществуват множество отделни стандартни библиотеки, оптимизирани за определена архитектура и операционна система.

В момента съществуват множество реализации на многонишкови библиотеки с различни цели. GNU Portable Threads (Pth) [1] е библиотека, създадена с идеята за интерфейс, който да е преносим за широк кръг UNIX системи. Библиотеката Next Generation POSIX Threading (NGPT) [4] е разработена от IBM с цел съвместимост с POSIX стандарта за Symmetric Multi-Processing (SMP) машини. Linux Threads [5] е реализация на POSIX IEEE 1003.1c стандарта за Linux платформи. Библиотеката е базирана на модел един-към-един и функционира в потребителски режим. Новата реализация на нишки в Linux – Native Posix Thread Library (NPTL) [6] е също съвместима с POSIX стандарта, но вече е базирана на kernel-space модела.

Реализациите са оптимизирани за различни архитектури машини-еднопроцесорни, мултипроцесорни с обща памет, мултипроцесорни с разпределена памет. Същевременно трябва да се отчита и факта, че не всеки клас от алгоритми подлежи на разпаралелване. Съществуват известни алгоритми, за които все още не е намерена ефективна многонишкова реализация, или дори е доказана невъзможността за създаване на такава. От друга страна, съществуват множество задачи, за които подобрението в бързината на изпълнение е голямо и оправдава допълнителните трудности, които се появяват при многопроцесорна работа. Разработчикът на програмно осигуряване трябва да има възможност за избор на различна функционалност на многонишкови библиотеки в зависимост от спецификата на различните задачи за решаване.

В настоящия доклад е представена спецификата на реализацията на многонишкова библиотека под Linux, която предоставя възможност за функциониране както в потребителски режим (user-space), така и в системен (kernel-space). Разработен е максимално идентичен потребителски интерфейс за реализациите в двата режима.

ОСОБЕНОСТИ НА РЕАЛИЗАЦИЯТА В USER-SPACE РЕЖИМ

Организация на превключването между нишките

Начинът на организиране на превключването на нишките от страна на диспечера има основно значение за ефективността на многонишковата библиотека.

Един възможен вариант за отнемане на процесора от потребителската нишка и предоставянето му на диспечера е използване на прекъсване. В Linux сигналите са удобна система за реализиране на такива потребителски софтуерни прекъсвания. Самото прекъсване може да се взима, например, от таймер настроен на определен интервал. Съществуват няколко проблема правещи такова решение не много желателно. На първо място, много голямо предимство на дизайна на библиотека изцяло в user-space, е възможността за бързо превключване между нишките. Използването на сигнали и таймери обаче изисква превключване към ядрото, което забавя многократно иначе тази бърза операция. На второ място, запазването на потребителския контекст, който е прекъснат от таймера, е трудно. Това е така, защото когато се изпълнява част от функция, обработваща прекъсване, за някои библиотечни функции не е гарантирано че функционират коректно.

Друг възможен подход е диспечерът да работи в отделна нишка, видима за операционната система, но това води до други трудности при реализацията без да предоставя нужната ефективност.

Това са основните причини, поради които се използва non-preemptive диспечеризиране в предложената реализация на user-space библиотеката. Това решение също предоставя възможност за ясно показване на основните предимства и недостатъци на създаването на нишки изцяло в user-space. Всяка нишка се представя чрез специална структура (Thread Control Block - TCB), съдържаща необходимата за управлението ѝ информация. Тази информация се използва от диспечера, реализиращ планирането и превключването на контекста. Той се грижи и за управлението на системата за събития, която е много важна за правилната работа на библиотеката. Диспечерът се извиква всеки път когато дадена нишка преотстъпва процесора на друга или когато се блокира автоматично към събитие.

Неблокиращите read/write операции, заспиване на нишка за определено време, операциите със синхронизиращите примитиви и функционалността за свързване на една нишка с друга са изцяло базирани на системата за събития.

ОСОБЕНОСТИ НА РЕАЛИЗАЦИЯТА В KERNEL-SPACE РЕЖИМ

Основната идея при този подход е да се използват средствата, налични на ниво операционна система, за диспечеризиране на задачи. Това дава възможност за много добро планиране и използване на ресурсите на цялата система, но недостатъкът е, че има по-бавно превключване между нишките поради нуждата от системни извиквания. За реализацията на нишките в този режим се използва системното извикване clone(). За ядрото на Linux нишка се съхранява в същата структура, в която се съхранява и отделен процес. Въпреки, че основната информация се съхранява в системните структури на операционната система, все пак е необходимо поддържането на информация за нишката в потребителския контекст. Такава информация, например, е за началната функция и нейния аргумент. Поддържането на такава дублирана информация позволява по-опростена реализация на някои от библиотечните функции.

В текущата реализация на библиотеката са разработени два типа примитиви за синхронизация – spinlocks и mutexes. Spinlocks предоставят синхронизация, базирана на активно изчакване. Въпреки, че този подход по принцип не е особено ефективен, реализацията е изключително опростена.

Вторият тип примитиви се базира на изчакване с освобождаване на процесора. Те са по-сложни за реализация, защото изискват използването на системно извикване към ядрото. Това е нужно, за да може диспечерът на операционната

система да преустанови временно изпълнението на нишката и да избере друга нишка, която да продължи своята работа. В предложената библиотека mutexes са реализирани по-възможно най-ефективния начин чрез смесен подход. Първоначално се прави опит за кратко активно изчакване. Ако след това mutex е все още зает, се преминава към блокиране на изпълнението на нишката от ядрото. За целта се използва предоставяното от Linux средство за базово заключване на достъпа Fast User-Space Mutex (futex) [2].

В общия случай за правилната работа на един futex е необходимо заемането на системен семафор. В текущата реализация се използват специални (Private) версии на futex, които са локални за процес. Така се спестява проверяването на семафора вътрешно в ядрото. Това предполага реализацията да е по-бърза от тази, предоставена от NPTL библиотеката, където mutex имат много по-сложна структура, забавяща тяхното използване.

ПРИЛОЖЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС

Библиотеката предоставя приложен потребителски интерфейс, който е максимално идентичен за реализациите в двата режима. Интерфейсът включва няколко групи типове данни и библиотечни функции с различно предназначение:

- Типове данни за работа със системни структури;
- Функции за инициализиране и завършване на работата с библиотеката;
- Функции за създаване и унищожаване на нишки;
- Функции за синхронизация;
- Функции за изчакване и блокировка;
- Функции за достъп до информация за отделна нишка.

ЕКСПЕРИМЕНТАЛНА ОЦЕНКА НА ФУНКЦИОНАЛНОСТТА НА БИБЛИОТЕКАТА

Тестването на многонишковата библиотека е направено по отношение на двата основни вида натоварване на системата:

- С процеси, ограничени по процесорно време. Това са процеси, които изпълняват голямо обем изчисления.
- С процеси, ограничени по входно-изходни операции. Това са процеси, които изпълняват интензивно системни извиквания към операционната система. Например, запис във файлове, очакване на вход от потребител, използване на мрежови комуникации.

Най-бързо изпълнение при процеси ограничени по процесорно време се очаква да бъде постигнато когато диспечерът използва максимално голям период между две превключвания на нишки. Това позволява запазване на информацията и максимално използване на кеш паметите на процесора. Тези процеси почти винаги работят без да се прекъсват, използвайки цялото им, определено от диспечера, време. Оптимална политика за тях е да им се позволява работа за голямо време, като при това не е необходимо честото им стартиране.

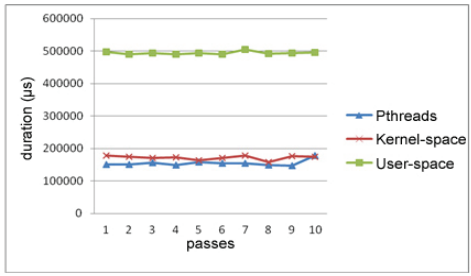
От друга страна, процесите, ограничени по входно изходни операции, не се нуждаят от дълги периоди, в които им се предоставя процесора. Това е така, защото те рядко го използват цялостно и често се блокират сами. Оптимална политика при тях е диспечерът да ги стартира колкото се може по-често за работа.

Направени са сравнения с най-разпространената многонишкова библиотека pthreads и по конкретно нейната най-последна реализация в Linux - NPTL.

Експерименталната платформа е базирана на Intel i7 2600K, работещ на 4.0GHz тактова честота. Процесорът използва “HT” технология и може да изпълнява 8 нишки едновременно. Всяко от ядрата има собствени кеш памети: L1 – 64KB, L2 – 256KB и общ L3 кеш с размер 8MB. Паметта е DDR3 SDRAM с размер 8GB,

1600MHz, двуканален режим. Операционната система е 64-битова дистрибуция Arch Linux с ядро версия 3.8, компилатор gcc 4.8.1 и glibc 2.17.

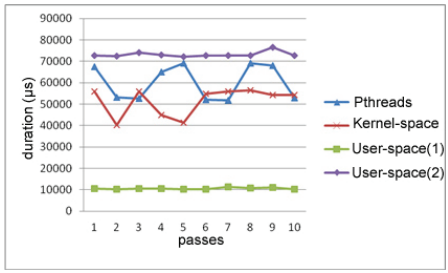
Първият тест включва умножение на матрици, като пример на задача, съдържаща множество изчисления. Създават се 5 отделни нишки, като всяка от тях умножава по 20 пъти квадратни матрици с размер 100x100 елемента. На фиг.1 са показани получените резултати.



Фиг. 1 – Резултати от тест с умножение на матрици

Очаквано pthreads и kernel-space реализациите се справят най-добре с теста. Резултатите са близки, но предимството остава за pthreads библиотеката. От този тест се вижда големият недостатък на user-space библиотеките – те могат да работят само на един процесор. Останалите две библиотеки се възползват от наличните 8 ядра. Тестът изключва наличие на блокиране на нишки.

Вторият тест използва работа с входно изходни операции. Създават се 2 нишки, едната от които чете от файл, а другата записва в него данни. Двойките операции запис/четене се извършват 1000000 пъти. Очаква се при някои от операциите да настъпват и блокирания. На фиг.2 са показани получените резултати.

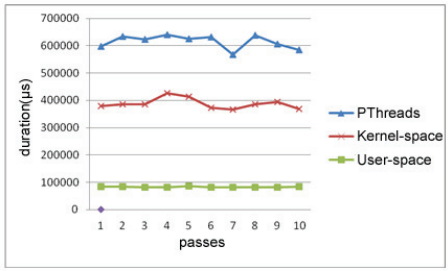


Фиг. 2 – Резултати от тест с входно-изходни операции

User-space библиотеката е използвана по два начина. С „1” е отбелязано използването на неблокиращи примитиви, предоставени от текущата реализация, а с „2” - когато те са заменени със стандартните write() и read(). Видна е по-добрата ефективност на разработените средства пред стандартните за Linux.

Библиотеките pthreads и kernel-space показват еднаква производителност с лек превес на kernel-space реализацията. При теста се наблюдава и по-голямо колебание между различните стартирания поради по-трудно предсказуемите варианти на изпълнение.

Третият тест цели оценка на ефективността на синхронизиращите примитиви. В цикъл от 1000000 итерации се заключва и се отключва mutex. Създават се 5 отделни нишки, всяка от които изпълнява описаното действие. Фигура 3 показва получените резултати.



Фиг. 3 – Резултати от тест със синхронизиращи примитиви

Вижда се, че реализация на mutex в библиотеката pthreads е по-бавна и изисква средно 615154μs.

Версията в kernel-space библиотеката е около 1.5 пъти по-бърза, благодарение на по-простата и ефективна структура на организацията на нишките. Също така предимството се дължи и на факта, че се използва краткотрайно активно изчакване, а не се преустановява веднага работата на нишката.

Съвсем очаквано user-space реализацията се оказва най-бърза със средно време от 82875μs. Това е поради факта, че не се достига до използване на бавни системни извиквания.

ЗАКЛЮЧЕНИЕ

В настоящия доклад е представена реализация на многонишкова библиотека под Linux, функционираща както в user-space, така и в kernel-space режим. Показани са някои характерни особености на реализацията. Направени са експериментални сравнения и оценки с известната библиотека pthreads.

Резултатите показват, че е постигнато идентично, в определени случаи и по-голямо бързодействие от съществуващи реализации. Разработената библиотека е сравнително малка по обем, което рефлектира в бърза компилация. Кодът е написан по опростен начин, което позволява използването на библиотеката при обучение по многонишково програмиране.

Като насоки за бъдеща работа се предвижда подобряване на обработката на сигнали в user-space режим, както и разработване на допълнителни синхронизиращи примитиви, като условни променливи и монитори.

ЛИТЕРАТУРА

[1] Engelschall R. Portable Multithreading. The signal stack trick for user-space thread creation. In Proceedings of 2000 USENIX Annual Technical Conference, 2000, pp.18-23.

[2] Futex Semantics and Usage. http://man7.org/linux/man-pages/man7/futex.7.html.

[3] Herlihy M. The Art of Multiprocessor Programming. Morgan Kaufmann, 2008.

[4] IBM Corporation. Next Generation POSIX Threading, November 2002.

[5] Leroy X. The LinuxThreads Library. http://pauillac.inria.fr/~xleroy/.

[6] Native Posix Thread Library. http://people.redhat.com/drepper/nptl-design.pdf.

За контакти:

доц. д-р инж. Христо Вълчанов, Катедра “Компютърни науки и технологии”, Технически университет-Варна, тел.: 052 383 424, e-mail: hristo@tu-varna.bg

Докладът е рецензиран.